SINGLE DEEP COUNTERFACTUAL REGRET MINIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Counterfactual Regret Minimization (CFR) is the most successful algorithm for finding approximate Nash equilibria in *imperfect information games*. However, CFR's reliance on full game-tree traversals limits its scalability and generality. Therefore, the game's state- and action-space is often *abstracted* (i.e. simplified) for CFR, and the resulting strategy is then mapped back to the full game. This requires extensive expert-knowledge, is not practical in many games outside of poker, and often converges to highly exploitable policies. A recently proposed method, *Deep CFR*, applies deep learning directly to CFR, allowing the agent to intrinsically abstract and generalize over the state-space from samples, without requiring expert knowledge. In this paper, we introduce *Single Deep CFR (SD-CFR)*, a variant of Deep CFR that has a lower overall approximation error by avoiding the training of an average strategy network. We show that SD-CFR is more attractive from a theoretical perspective and empirically outperforms Deep CFR with respect to exploitability and one-on-one play in poker.

1 INTRODUCTION

In perfect information games, players usually seek to play an optimal deterministic strategy. In contrast, sound policy optimization algorithms for imperfect information games converge towards a *Nash equilibrium*, a distributional strategy characterized by minimizing the losses against a worst-case opponent. The most popular family of algorithms for finding such equilibria is Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2008). Conventional CFR methods iteratively traverse the game-tree to improve the strategy played in each state. For instance, CFR$^+$ (Tammelin, 2014), a fast variant of CFR, was used to solve two-player Limit Texas Hold'em Poker (Bowling et al., 2015; Tammelin et al., 2015), a variant of poker frequently played by humans.

However, the scalability of such *tabular* CFR methods is limited since they need to visit a given state to update the policy played in it. In games too large to fully traverse, practitioners hence often employ domain-specific abstraction schemes (Ganzfried & Sandholm, 2014; Brown et al., 2015) that can be mapped back to the full game after training has finished. Unfortunately, these techniques have been shown to lead to highly exploitable policies in the large benchmark game Heads-Up No-Limit Texas Hold'em Poker (HUNL) (Lisy & Bowling, 2016) and typically require extensive expert knowledge. To address these two problems, researchers started to augment CFR with neural network function approximation, first resulting in DeepStack (Moravčík et al., 2017). Concurrently with Libratus (Brown & Sandholm, 2018a), DeepStack was one of the first algorithms to defeat professional poker players in HUNL, a game consisting of 10^{160} states and thus being far too large to fully traverse.

While tabular CFR has to visit a state of the game to update its policy in it, a parameterized policy may be able to play an educated strategy in states it has never seen before. Purely parameterized (i.e. non-tabular) policies have led to great breakthroughs in AI for perfect information games (Mnih et al., 2015; Schulman et al., 2017; Silver et al., 2017) and were recently also applied to large imperfect information games by Deep CFR (Brown et al., 2018a) to mimic a variant of tabular CFR from samples.

Deep CFR's strategy relies on a series of two independent neural approximations. In this paper, we introduce *Single Deep CFR (SD-CFR)*, a simplified variant of Deep CFR that obtains its final strategy after just one neural approximation by using what Deep CFR calls *value networks* directly instead of training an additional network to approximate the weighted average strategy. This reduces the overall

sampling- and approximation error and makes training more efficient. We show experimentally that SD-CFR improves upon the convergence of Deep CFR in poker games and outperforms Deep CFR in one-one-one matches.

2 EXTENSIVE-FORM GAMES

This section introduces extensive-form games and the notation we will use throughout this work. Formally, a finite two-player extensive-form game with imperfect information is a set of **histories** $\mathcal{H}$, where each history is a path from the root $\phi \in \mathcal{H}$ to any particular state. The subset $\mathcal{Z} \subset \mathcal{H}$ contains all terminal histories. $A(h)$ is the set of actions available to the acting player at history h, who is chosen from the set $\{1, 2, \text{chance}\}$ by the player function $P(h)$. In any $h \in \mathcal{H}$ where $P(h) = chance$, the action is chosen by the dynamics of the game itself. Let $N = \{1, 2\}$ be the set of both players. When referring to a player $i \in N$, we refer to his opponent by $-i$. All nodes $z \in \mathcal{Z}$ have an associated **utility** $u(z)$ for each player. This work focuses on **zero-sum** games, defined by the property $u_i(z) = -u_{-i}(z)$ for all $z \in \mathcal{Z}$.

Imperfect information is represented by partitioning $\mathcal{H}$ into **information sets**. An information set I_i is a subset of $\mathcal{H}$, where histories $h, h' \in \mathcal{H}$ are in the same information set if and only if player i cannot distinguish between h and h' given his private and all available public information. For each player $i \in N$, an **information partition** $\mathcal{I}_i$ is a set of all such information sets. Let $A(I) = A(h)$ and $P(I) = P(h)$ for all $h \in I$ and each $I \in \mathcal{I}_i$.

Each player i chooses actions according to a **behavioural strategy** σ_i, with $\sigma_i(I, a)$ being the probability of choosing action a when in I. We refer to a tuple (σ_1, σ_2) as a **strategy profile** σ. Let $\pi^\sigma(h)$ be the probability of reaching history h if both players follow σ and let $\pi_i^\sigma(h)$ be the probability of reaching h if player i acts according to σ_i and player $-i$ always acts deterministically to get to h. It follows that the probability of reaching an information set I if both players follow σ is $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$ and is $\pi_i^\sigma(I) = \sum_{h \in I} \pi_i^\sigma(h)$ if $-i$ plays to get to I.

Player i's **expected utility** from any history h assuming both players follow strategy profile σ from h onward is denoted by $u_i^\sigma(h)$. Thus, their expected utility over the whole game given a strategy profile σ can be written as $u_i^\sigma(\phi) = \sum_{z \in \mathcal{Z}} \pi^\sigma(z) u_i(z)$.

Finally, a strategy profile $\sigma = (\sigma_1, \sigma_2)$ is a **Nash equilibrium** if no player i could increase their expected utility by deviating from σ_i while $-i$ plays according to σ_{-i}. We measure the **exploitability** $e(\sigma)$ of a strategy profile by how much its optimal counter strategy profile (also called **best response**) can beat it by. Let us denote a function that returns the best response to σ_i by $BR(\sigma_i)$. Formally,

$$e(\sigma) = -\sum_{i \in N} (u_i(\sigma_i, BR(\sigma_i))$$

3 COUNTERFACTUAL REGRET MINIMIZATION (CFR)

Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2008) is an iterative algorithm. It can run either *simultaneous* or *alternating* updates. If the former is chosen, CFR produces a new *iteration-strategy* σ_i^t for all players $i \in N$ on each iteration t. In contrast, alternating updates produce a new strategy for only one player per iteration, with player $t \bmod 2$ updating his on iteration t.

To understand how CFR converges to a Nash equilibrium, let us first define the *instantaneous regret* for player i of action $a \in A(I)$ in any $I \in \mathcal{I}_i$ as

$$r_i^t(I, a) = \pi_{-i}^{\sigma^t}(I)(v_i^{\sigma^t}(I, a) - v_i^{\sigma^t}(I)) \tag{1}$$

where $v_i^{\sigma^t}(I) = \sum_{h \in I} \frac{\pi_{-i}^{\sigma^t}(h) u_i^{\sigma^t}(h)}{\pi_{-i}^{\sigma^t}(I)}$ and $v_i^{\sigma^t}(I, a) = \sum_{h \in I} \frac{\pi_{-i}^{\sigma^t}(h) u_i^{\sigma^t}(h \xrightarrow{act} a)}{\pi_{-i}^{\sigma^t}(I)}$. Intuitively, $r_i^t(I, a)$ quantifies how much more player i would have won (in expectation), had he always chosen a in I and played to get to I but according to σ^t thereafter. The *overall regret* on iteration T is $R_i^T(I, a) = \sum_{t=1}^T r_i^t(I, a)$. Now, the iteration-strategy for player i can be derived by

$$\sigma_i^{t+1}(I, a) = \begin{cases} \frac{R_i^t(I,a)_+}{\sum_{\tilde{a} \in A(I)} R_i^t(I,\tilde{a})_+} & \text{if} \sum_{\tilde{a} \in A(I)} R_i^t(I, \tilde{a})_+ > 0 \\ \frac{1}{|A(I)|} & \text{otherwise} \end{cases} \tag{2}$$

where $x_+ = max(x, 0)$. Note that $\sigma_i^0(I, a) = \frac{1}{|A(I)|}$.

The iteration-strategy profile σ^t does not converge to an equilibrium as $t \to \infty$ in most variants of CFR [1]. The policy that has been shown to converge to an equilibrium profile is the *average strategy* $\bar{\sigma}_i^T$. For all $I \in \mathcal{I}_i$ and each $a \in A(I)$ it is defined as

$$\bar{\sigma}_i^T(I, a) = \frac{\sum_{t=1}^T \pi_i^{\sigma^t}(I)\sigma_i^t(I, a)}{\sum_{t=1}^T \pi_i^{\sigma^t}(I)} \tag{3}$$

3.1 VARIATIONS OF CFR

Aiming to solve ever bigger games, researchers have proposed many improvements upon vanilla CFR over the years (Tammelin et al., 2015; Brown & Sandholm, 2018a; Moravčík et al., 2017). These improvements include alternative methods for regret updates (Tammelin, 2014; Brown & Sandholm, 2018b), automated schemes for abstraction design (Ganzfried & Sandholm, 2014), and sampling variants of CFR (Lanctot et al., 2009). Many of the most successful algorithms of the recent past also employ *real-time solving* or *re-solving* (Brown et al., 2018b; Moravčík et al., 2017).

Discounted CFR (DCFR) (Brown & Sandholm, 2018b) slightly modifies the equations for $R_i^T(I, a)$ and $\bar{\sigma}_i^T$. A special case of DCFR is *linear CFR (LCFR)*, where the contribution of the instantaneous regret of iteration t as well as the contribution of σ^t to $\bar{\sigma}^T$ is weighted by t. This change alone suffices to let LCFR converge up to two orders of magnitude faster than vanilla CFR does in some large games.

Monte-Carlo CFR (MC-CFR) (Lanctot et al., 2009) proposes a family of tabular methods that visit only a subset of information sets on each iteration. Different variants of MC-CFR can be constructed by choosing different sampling policies. One such variant is *External Sampling (ES)*, which executes all actions for player i, the traverser, in every $I \in \mathcal{I}_i$ but draws only one sample for actions not controlled by i (i.e. those of $-i$ and chance). In games with many player-actions *Average Strategy Sampling* (Burch et al., 2012), *Robust Sampling* (Hui et al., 2018) are very useful. They, in different ways, sample only a sub-set of actions for i. Both LCFR and a similarly fast variant called CFR$^+$ (Tammelin, 2014) are compatible with forms of MC sampling, although CFR$^+$ was regarded as to sensitive to variance until recently (Schmid et al., 2018).

4 DEEP CFR

CFR methods either need to run on the full game tree or employ domain-specific abstractions. The former is infeasible in large games and the latter not easily possible in all domains. Deep CFR (Brown et al., 2018a) computes an approximation of linear CFR (Brown & Sandholm, 2018b) with alternating player updates. It is sample-based and does not need to store regret tables, making it generally applicable to any two-player zero-sum game.

On each iteration, Deep CFR fits a *value network* $\hat{D}_i$ for one player i to approximate what we call *advantage*, which is defined as $D_i^T(I, a) = \frac{R_{i,linear}^T(I,a)}{\sum_{t=1}^T(t\pi_{-i}^{\sigma^t}(I))}$, where $R_{i,linear}^T(I, a) = \sum_{t=1}^T(tr_i^t(I, a))$.

In large games, reach-probabilities naturally are (on average) very small after many tree-branchings. Considering that it is hard for neural networks to learn values across many orders of magnitude (van Hasselt et al., 2016), Deep CFR divides $R_{i,linear(I,a)}^T$ by the total linear reach $\sum_{t=1}^T(t\pi_{-i}^{\sigma^t}(I))$ and thereby avoids this problem. This still yields correct results because $\sum_{t=1}^T(t\pi_{-i}^{\sigma^t}(I))$ is identical for all $a \in A(I)$.

We can derive the iteration-strategy for $t + 1$ from D^t similarly to CFR in equation 2 by

$$\sigma_i^{t+1}(I, a) = \begin{cases} \frac{D_i^t(I,a)_+}{\sum_{\tilde{a}\in A(I)} D_i^t(I,\tilde{a})_+} & \text{if} \sum_{\tilde{a}\in A(I)} D_i^t(I, \tilde{a})_+ > 0 \\ \frac{1}{|A(I)|} & \text{otherwise} \end{cases} \tag{4}$$

[1] In CFR-BR (Johanson et al., 2012) σ^t does converge probabilistically as $t \to \inf$ and in CFR$^+$ (Tammelin, 2014) it often does so empirically (but without guarantees); in vanilla CFR and linear CFR (Brown & Sandholm, 2018b) σ^t typically does not converge.

However, Deep CFR modifies this to heuristically choose the action with the highest advantage whenever $\sum_{\tilde{a} \in A(I)} D_i^t(I, \tilde{a})_+ \leq 0$. Deep CFR obtains the training data for $\hat{D}$ via batched external sampling (Lanctot et al., 2009; Brown et al., 2018a). All instantaneous regret values collected over the N traversals are stored in a memory buffer B_i^v. After its maximum capacity is reached, B_i^v is updated via reservoir sampling (Vitter, 1985). To mimic the behaviour of linear CFR, we need to weight the training losses between the predictions $\hat{D}$ makes and the sampled regret vectors in B_i^v with the iteration-number on which a given datapoint was added to the buffer.

At the end of its training procedure (i.e. after the last iteration), Deep CFR fits another neural network $\hat{S}_i(I, a)$ to approximate the linear average strategy

$$\bar{\sigma}_i^T(I, a) = \frac{\sum_{t=1}^T (t \pi_i^{\sigma^t}(I) \sigma_i^t(I, a))}{\sum_{t=1}^T (t \pi_i^{\sigma^t}(I))} \tag{5}$$

Data to train $\hat{S}_i$ is collected in a separate reservoir buffer B_i^s during the same traversals that data for B_i^v is being collected on. Recall that external sampling always samples all actions for the traverser, let us call him i, and plays according to σ_{-i}^t for the opponent. Thus, when i is the traverser, $-i$ is the one who adds his strategy vector $\sigma_{-i}^t(I)$ to B_{-i}^s in every $I \in \mathcal{I}_{-i}$ visited during this traversal. This elegantly assures that the likelihood of $\sigma_{-i}^t(I)$ being added to B_{-i}^s on any given traversal is proportional to $\pi_{-i}^{\sigma^t}(I)$. Like before, we also need to weight the training loss for each datapoint by the iteration-number on which the datapoint was created.

Notice that tabular CFR achieves importance weighting between iterations through multiplying with some form of the reach probability (see equations 1 and 3). In contrast, Deep CFR does so by controlling the expected frequency of datapoints from different iterations occurring in its buffers and by weighting the neural network losses differently for data from each iteration.

5 Single Deep Counterfactual Regret Minimization (SD-CFR)

Notice that storing all iteration-strategies would allow one to compute the average strategy on the fly *during play* both in tabular and approximate CFR variants. In tabular methods, the gain of not needing to keep $\bar{\sigma}$ in memory during training would come at the cost of storing t equally large tables (though potentially on disk) during training and during play. However, this is very different with Deep CFR. Not aggregating into $\hat{S}$ removes the sampling- and approximation error that B^s and $\hat{S}$ introduce, respectively. Moreover, the computational work needed to train $\hat{S}$ is no longer required. Like in the tabular case, we do need to keep all iteration strategies, but this is much cheaper with Deep CFR as strategies are compressed within small neural networks.

We will now look at two methods for querying $\bar{\sigma}$ from a buffer of past value networks B^M.

5.1 Acting on freely playable trajectories

Often (e.g. in one-one-one evaluations and during rollouts), a trajectory is played from the root of the game-tree and the agent is only required to return action-samples of the average strategy on each step forward. In this case, SD-CFR chooses a value network $\hat{D}_i^t \in B_i^M$ at the start of the game, where each $\hat{D}_i^t$ is assigned sampling weight t. The policy σ_i, which this network gives by equation 4, is now going to be used for the whole game trajectory. We call this method *trajectory-sampling*.

By applying the sampling weights when selecting a $\hat{D}_i \in B_i^M$, we satisfy the linear averaging constraint of equation 5, and by using the same σ_i for the whole trajectory starting at the root, we ensure that the iteration-strategies are also weighted proportionally to each of their reach-probabilities in any given state along that trajectory. The latter happens naturally, since $\hat{D}_i^t$ of any t produces σ_i^t, which reaches each information set I with a likelihood directly proportional to $\pi_i^{\sigma^t}(I)$ when playing from the root.

The query cost of this method is constant with the number of iterations (and equal to the cost of querying Deep CFR).

5.2 QUERYING A COMPLETE ACTION DISTRIBUTION IN ANY INFORMATION SET

Let us now consider querying the complete action probability distribution $\bar{\sigma}_i^T(I)$ in some information set $I \in \mathcal{I}_i$. Given B_i^M, we can compute $\bar{\sigma}_i^T(I)$ exactly through equation 5, where we compute

$$\pi_i^{\sigma^t}(I) = \prod_{I' \in I, P(I')=i, a':I' \to I} \sigma_i^t(I', a') \tag{6}$$

Here, $I' \in I$ means that I' is on the trajectory leading to I and $a' : I' \to I$ is the action selected in I' leading to I.

This computation can be done with at most[2] as many feedforward passes through each network in B_i^M as player i had decisions along the trajectory to I, typically taking a few seconds in poker when done on a CPU.

5.3 QUERYING A COMPLETE ACTION DISTRIBUTION ON A TRAJECTORY

If a trajectory is played forward from the root, as is the case in e.g. exploitability evaluation, we can cache the step-wise reach-probabilities on each step I^k along the trajectory and compute $\pi_i^{\sigma^t}(I^{k+1}) = \sigma_i^t(I^{k+1}, a')\pi_i^{\sigma^t}(I^k)$, where a' is the action that leads from I^k to I^{k+1}. This reduces the number of queries per step to at most $|B_i^M|$.

5.4 THEORETICAL AND PRACTICAL PROPERTIES

SD-CFR always mimics $\bar{\sigma}_i^T$ correctly from the iteration-strategies it is given. Thus, if these iteration-strategies were perfect approximations of the real iteration-strategies, SD-CFR is equivalent to linear CFR (see Theorem 2), which is not necessarily true for Deep CFR (see Theorem 1).

As we later show in an experiment, SD-CFR's performance degrades if reservoir sampling is performed on B^M after the number of iterations trained exceeds the buffer's capacity. Thankfully, the neural network proposed to be used for Deep CFR in large poker games has under 100,000 parameters (Brown et al., 2018a) and thus requires under 400KB of disk space. Deep CFR is usually trained for just a few hundred iterations (Brown et al., 2018a), but storing even 25,000 such networks on disk would need only 10GB of disk space. At no point during any computation do we need all networks in memory. Thus, keeping all value networks will not represent a problem in practise.

Observing that Deep CFR and SD-CFR depend upon the accuracy of the value networks in exactly the same way, we can conclude that SD-CFR is a better or equally good approximation of linear CFR as long as all value networks are stored. Though this shows that SD-CFR is largely superior in theory, it is not implicit that SD-CFR will always produce stronger strategies empirically. We will investigate this next.

Theorem 1. *If the capacity of strategy buffer B_i^s is finite or if only a finite number K of traversals is executed per iteration, B_i^s is not guaranteed to reflect the true average strategy $\bar{\sigma}_i^T(I)$ for every $I \in \mathcal{I}_i$ even if all value networks are perfect approximators of the true advantage after any number of training iterations $T > 2$. Hence, even a perfect function approximator for $\hat{S}$ is not guaranteed to model $\bar{\sigma}_i^T$ without error.*

Theorem 2. *Assume that for all $i \in N$, all $I \in \mathcal{I}_i$, all $a \in A(I)$, and all t up to the number of iterations trained T, $\hat{D}_i^t(I, a) = D_i^t(I, a)$ (i.e. that all value networks perfectly model the true advantages). Now, SD-CFR represents $\bar{\sigma}_i^T$ without error. This holds for both trajectory-sampling SD-CFR and for when SD-CFR computes $\bar{\sigma}_i^T(I)$ explicitly. Furthermore, an opponent has no way of distinguishing which of the two proposed methods of sampling from $\bar{\sigma}$ is used solely from gameplay.*

Proofs for both Theorem 1 and 2 can be found in the supplementary material.

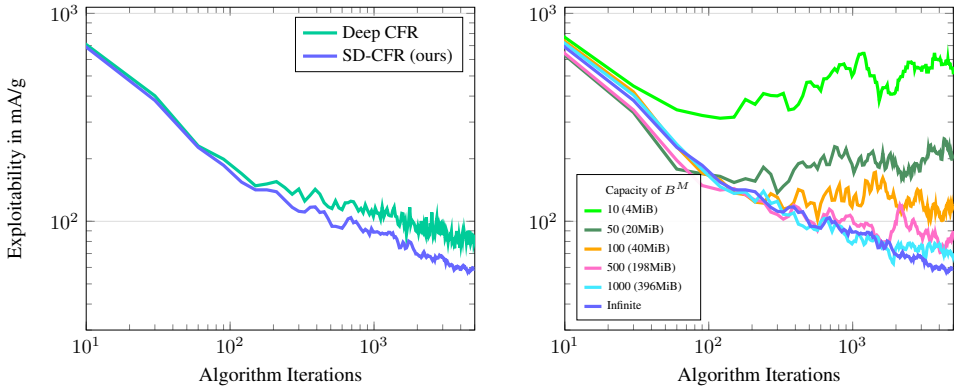
[2]This number can further be reduced by omitting queries for any σ^t as soon as it assigns probability 0 to the action played on the trajectory.

6 EXPERIMENTS

We empirically evaluate SD-CFR by comparing to Deep CFR and by analyzing the effect of sampling on B^M. Recall that Deep CFR and SD-CFR are equivalent in how they train their value networks. This allows both algorithms to *share the same value networks* in our experiments, which makes comparisons far less susceptible to variance over algorithm runs and conveniently guarantees that both algorithms tend to the same Nash equilibrium.

Where not otherwise noted, we use hyperparamters as Brown et al. (2018a). Our environment observations include additional features such as the size of the pot and represent cards as concatenated one-hot vectors without any higher level features, but are otherwise as Brown et al. (2018a).

6.1 EXPLOITABILITY IN LEDUC POKER



(a) Comparing SD-CFR and Deep CFR

(b) Sampling on B^M with finite capacity

Figure 1: Empirical analysis of SD-CFR in Leduc Hold'em Poker. Results in Figure 1a and 1b are averaged over five and three runs, respectively.

Figure 1a shows the exploitability (i.e. loss against a worst-case opponent) of SD-CFR and Deep CFR in Leduc Poker (Southey et al., 2005) measured in *milli-antes per game (mA/g)*.

In Leduc Poker, players start with an infinite number of chips. The deck has six cards of two suits $\{a, b\}$ and three ranks $\{J, Q, K\}$. There are two betting rounds: preflop and flop. After the preflop, a card is publicly revealed. At the start of the game, each player adds 1 chip, called the ante, to the pot and is dealt one private card. There are at most two bets/raises per round, where the bet-size is fixed at 2 chips in the preflop, and 4 chips after the flop is revealed. If no player folded, the winner is determined via hand strength. If a player made a pair with the public card, they win. Otherwise $K > Q > J$. If both players hold a card of the same rank, the pot is split.

Hyperparameters are chosen to favour Deep CFR as the neural networks and buffers are very large in relation to the size of the game. Yet, we find that SD-CFR minimizes exploitability better than Deep CFR. Exact hyperparameters can be found in the supplementary material.

Although we concluded that storing all value networks is feasible, we analyze the effect of reservoir sampling on B^M in Figure 1b and find it leads to plateauing and oscillation, at least up to $|B^M| = 1000$.

6.2 ONE-ON-ONE MATCHES IN 5-FLOP HOLD'EM POKER (5-FHP) AGAINST DEEP CFR

Figure 2 shows the results of one-one-one matches between SD-CFR and Deep CFR in 5-Flop Hold'em Poker (5-FHP). 5-FHP is a large poker game similar to regular FHP (Brown et al., 2018a), which was used to evaluate Deep CFR (Brown et al., 2018a). The only difference is that 5-FHP uses five instead of three flop cards, forcing agents to abstract and generalize more. For details on

DEPTH	ROUND	DIF MEAN	DIF STD	N
0	PF	0.012± 0.0001	0.017	200K
1	PF	0.013± 0.0001	0.018	100K
2	FL	0.052± 0.0003	0.048	80K
3	FL	0.083± 0.0005	0.075	83K
4	FL	0.113± 0.0011	0.109	37K
5	FL	0.175± 0.0057	0.206	5K

Table 1: **Disagreement between SD-CFR's and Deep CFR's average strategies**. "DEPTH": number of player actions up until the measurement, "ROUND": PF=Preflop, FL=Flop, "DIF MEAN": mean and 95% confidence interval of the absolute differences between the strategies over the "N" occurrences. "DIF STD": approximate standard deviation of agreement across information sets.

FHP, please refer to (Brown et al., 2018a). The neural architecture is as Brown et al. (2018a). Both algorithms again *share the same value networks* during each training run. Like Brown et al. (2018a), B^v and B^s have a capacity of 40 million per player. On each iteration, we run a batch of 300,000 external sampling traversals and train a value network from scratch using a batch size of 10,240 for 4,000 updates. Average strategy networks are trained with a batch size of 20,480 for 20,000 updates. SD-CFR's B^M stores all value networks, requiring 120MB of disk space, while each B^s needs around 25GB of memory during training.

The y-axis plots SD-CFR's average winnings against Deep CFR in *milli-big blinds per game (mbb/g)* measured every 30 iterations. For reference, 10 mbb/g is considered a good margin between humans in Heads-Up Limit Hold'em (HULH), a game with longer action sequences, but similar minimum and maximum winnings per game as 5-FHP. Measuring the performance on iteration t compares how well the SD-CFR averaging procedure would do against the one of Deep CFR if the algorithm stopped training after t iterations

B^s reached its maximum capacity of 40 million for both players by iteration 120 in all runs. Before this point, SD-CFR defeats Deep CFR by a sizable margin, but even after that, SD-CFR clearly defeats Deep CFR.

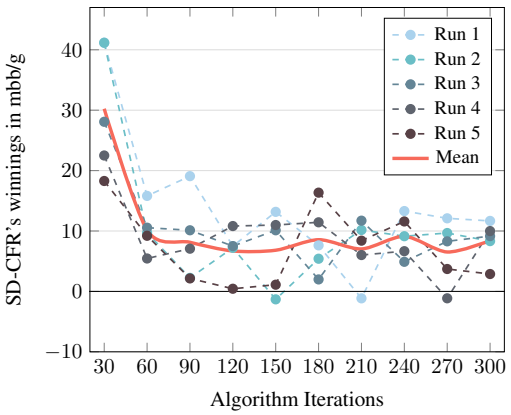


Figure 2: **One-on-One performance** of Single Deep CFR vs. Deep CFR. Dashed lines are independent algorithm runs. All evaluations have 95% confidence intervals between ±5.4 and ±6.51 and are the average result of 3M poker hands each.

6.2.1 COMPARING STRATEGIES

We analyze how far the average strategies of SD-CFR and Deep CFR are apart at different depths of the tree of 5-FHP. In particular, we measure

$$\frac{1}{2}\sum_{i\in 1,2}\left(\mathbb{E}_{I_i\sim\bar{\sigma}_i^T}\sum_{a\in A(I)}|\bar{\sigma}_i^{T,SD}(I,a)-\bar{\sigma}_i^{T,\hat{S}}(I,a)|\right)$$

We ran 200,000 trajectory rollouts for each player, where player i plays according to SD-CFR's average strategy $\bar{\sigma}_i^{T,SD}$ and $-i$ plays uniformly random. Hence, we only evaluate on trajectories on which the agent should feel comfortable. The two agents again share the same value networks and thus approximate the same equilibrium. We trained for 180 iterations, a little more than it takes for B^s and B^v to be full for both players. Table 1 shows that Deep CFR's approximation is good on early levels of the tree but has a larger error in information sets reached only after multiple decision points.

7 RELATED WORK

Regression CFR (R-CFR) (Waugh et al., 2015) applies regression trees to estimate regret values in CFR and CFR$^+$. Unfortunately, despite promising expectations, recent work failed to apply R-CFR in combination with sampling (Srinivasan et al., 2018). *Advantage Regret Minimization (ARM)* (Jin et al., 2017) is similar to R-CFR but was only applied to single-player environments. Nevertheless, ARM did show that regret-based methods can be of interest in imperfect information games much bigger, less structured, and more chaotic than poker.

DeepStack (Moravčík et al., 2017) was the first algorithm to defeat professional poker players in one-on-one gameplay of Heads-Up No-Limit Hold'em Poker (HUNL) requiring just a single GPU and CPU for real-time play. It accomplished this through combining real-time solving with counterfactual value approximation with deep networks. Unfortunately, DeepStack relies on tabular CFR methods without card abstraction to generate data for its counterfactual value networks, which could make applications to domains with many more private information states than HUNL has difficult.

Neural Fictitious Self-Play (NFSP) (Heinrich & Silver, 2016) was the first algorithm to soundly apply deep reinforcement learning from single trajectory samples to large extensive-form games. While not showing record-breaking results in terms of exploitability, NFSP was able to learn a competitive strategy in Limit Texas Hold'em Poker over just 14 GPU/days. Recent literature elaborates on the convergence properties of multi-agent deep reinforcement learning (Lanctot et al., 2017) and introduces novel actor-critic algorithms (Srinivasan et al., 2018) that have similar convergence properties as NFSP and SD-CFR.

8 FUTURE WORK

So far, Deep CFR was only evaluated in games with three player actions. Since external sampling would likely be intractable in games with tens or more actions, one could employ outcome sampling (Lanctot et al., 2009), robust sampling (Hui et al., 2018), Targeted CFR (Jackson, 2017), or average-strategy-sampling (Burch et al., 2012) in such settings.

To avoid action translation after training in an action-abstracted game, continuous approximations of large discrete action-spaces where actions are closely related (e.g. bet-size selection in No-Limit Poker games, auctions, settlements, etc.) could be of interest. This might be achieved by having the value networks predict parameters to a continuous function whose integral can be evaluated efficiently. The iteration-strategy could be derived by normalizing the advantage clipped below 0. The probability of action a could be calculated as the integral of the strategy on the interval corresponding to a in the discrete space.

Given a few modifications to its neural architecture and sampling procedure, SD-CFR could potentially be applied to much less structured domains than poker such as those that deep reinforcement learning methods like PPO (Schulman et al., 2017) are usually applied to. A first step on this line of research could be to evaluate whether SD-CFR is preferred over approaches such as (Srinivasan et al., 2018) in these settings.

9 CONCLUSIONS

We introduced *Single Deep CFR (SD-CFR)*, a new variant of CFR that uses function approximation and partial tree traversals to generalize over the game's state space. In contrast to previous work, SD-CFR extracts the average strategy directly from a buffer of value networks from past iterations. We show that SD-CFR is more attractive in theory and performs much better in practise than Deep CFR.

ACKNOWLEDGMENTS

REFERENCES

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018a.

Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. *arXiv preprint arXiv:1809.04040*, 2018b.

Noam Brown, Sam Ganzfried, and Tuomas Sandholm. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit texas hold'em agent. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 7–15. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. *arXiv preprint arXiv:1811.00164*, 2018a.

Noam Brown, Tuomas Sandholm, and Brandon Amos. Depth-limited solving for imperfect-information games. *arXiv preprint arXiv:1805.08195*, 2018b.

Neil Burch, Marc Lanctot, Duane Szafron, and Richard G Gibson. Efficient monte carlo counterfactual regret minimization in games with many player actions. In *Advances in Neural Information Processing Systems*, pp. 1880–1888, 2012.

Sam Ganzfried and Tuomas Sandholm. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In *AAAI*, pp. 682–690, 2014.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Li Hui, Hu Kailiang, Ge Zhibang, Jiang Tao, Qi Yuan, and Song Le. Double neural counterfactual regret minimization. *https://openreview.net/pdf?id=Bkeuz20cYm, 2018*, 2018.

Eric Griffin Jackson. Targeted cfr. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Peter H Jin, Sergey Levine, and Kurt Keutzer. Regret minimization for partially observable deep reinforcement learning. *arXiv preprint arXiv:1710.11424*, 2017.

Michael Johanson, Nolan Bard, Neil Burch, and Michael Bowling. Finding optimal abstract strategies in extensive-form games. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, pp. 1078–1086, 2009.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4190–4203, 2017.

Viliam Lisy and Michael Bowling. Equilibrium approximation quality of current no-limit poker bots. *arXiv preprint arXiv:1612.07547*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Martin Schmid, Neil Burch, Marc Lanctot, Matej Moravcik, Rudolf Kadlec, and Michael Bowling. Variance reduction in monte carlo counterfactual regret minimization (vr-mccfr) for extensive form games using baselines. *arXiv preprint arXiv:1809.03057*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Finnegan Southey, Michael P Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pp. 550—-558, 2005.

Sriram Srinivasan, Marc Lanctot, Vinicius Zambaldi, Julien Pérolat, Karl Tuyls, Rémi Munos, and Michael Bowling. Actor-critic policy optimization in partially observable multiagent environments. In *Advances in Neural Information Processing Systems*, pp. 3426–3439, 2018.

Oskari Tammelin. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*, 2014.

Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit texas hold'em. In *International Joint Conference on Artificial Intelligence*, pp. 645–652, 2015.

Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems*, pp. 4287–4295, 2016.

Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

Kevin Waugh, Dustin Morrill, James Andrew Bagnell, and Michael Bowling. Solving games with functional regret estimation. In *Association for the Advancement of Artificial Intelligence*, volume 15, pp. 2138–2144, 2015.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pp. 1729–1736, 2008.

A HYPERPARAMETERS OF EXPERIMENTS PERFORMED IN LEDUC HOLD'EM POKER

B^v and B^s have a capacity of 1 million for each player. On each iteration, data is collected over 1,500 external sampling traversals and a new value network is trained to convergence (750 updates of batch size 2048), initialized randomly at $t < 2$ and with the weights of the value net from iteration $t - 2$ afterwards. Average-strategy networks are trained to convergence (5000 updates of batch size 2048) always from a random initialization. All networks used for this evaluation have 3 fully-connected layers of 64 units each, which adds up to more parameters than Leduc Hold'em has states. All other hyperparameters were chosen as in (Brown et al., 2018a).

B RULES OF LEDUC HOLD'EM POKER

Leduc Hold'em Poker is a two-player game, were players alternate seats after each round. At the start of the game, both players add 1 chip, the *ante*, to the pot and are dealt a *private card* (unknown to the opponent) from a deck consisting of 6 cards: {A, A, B, B, C, C}. There are two rounds: *pre-flop* and *flop*. The game starts at the pre-flop and transitions to the flop after both players have acted and wagered the same number of chips. At each decision point, players can choose an action from a subset of {*fold*,*call*, *raise*}. When a player folds, the game ends and all chips in the pot are awarded to the opponent. Calling means matching the opponent's raise. The first player to act in a round has the option of *checking*, which is essentially a call of zero chips. Their opponent can then bet or also check. When a player raises, he adds more chips to the pot than his opponent wagered so far. In Leduc Hold'em, the number of raises per round is capped at 2. Each raise adds 2 additional chips in the pre-flop round and 4 in the flop round. On the transition from pre-flop to flop, one card from the remaining deck is revealed publicly. If no player folded and the game ends with a player calling, they show their hands and determine the winner by the rule that if a player's private card matches the flop card, they win. Otherwise the player with the higher card according to $A\ B\ C$ wins.

C PROOF OF THEOREM 1

Proof. Let I be any information set in $\mathcal{I}_i$. Assuming that $0 < \pi_i^{\sigma^t}(I) < 1$. Recall that external sampling samples only one action for player i and $chance$ at any decision point, when $-i$ is the traverser. Since $(1 - \pi_i^{\sigma^t}(I))^K > 0$ for any finite number of traversals K per iteration, we cannot guarantee that I will be visited. If I is not visited despite $\pi_i^{\sigma^t}(I) > 0$, the contribution of σ_i^t to $\bar{\sigma}_i^T(I)$ is not represented in B_i^s.

For the second argument, we assume that $K = \infty$. Let I again be any information set in $\mathcal{I}_i$ in which $|A(I)| > 1$. Assume that $\pi_i^{\sigma^t}(I)$ is irrational and that $\pi_i^{\sigma^j}(I)$ is rational. Clearly, because its capacity is finite, B_i^s could not reflect the ratio between $\pi_i^{\sigma^t}(I)$ and $\pi_i^{\sigma^j}(I)$ correctly through the frequency of the appearance of samples from iterations t and j, regardless of the number of traversals. Furthermore, in games where the number of members in the set

$$\{I \in \mathcal{I}_i : |A(I)| > 1, \pi_i^{\sigma^t}(I) > 0\}$$

is bigger than the capacity of B_i^s, not every $I \in \tilde{I}$ can fit into B_i^s on iteration t, also making B_i^s an incomplete representation of $\bar{\sigma}_i^T(I)$. □

D PROOF OF THEOREM 2

Proof. Let B_i^M be a buffer of all value networks up to iteration T belonging to player i.

Since $\hat{D}_i^t(I, a) = D_i^t(I, a)$ for all $I \in \mathcal{I}_i$ and all $a \in A(I)$ by assumption,

$$\sigma_i^{t+1}(I, a) = \begin{cases} \frac{D_i^t(I,a)_+}{\sum_{\tilde{a} \in A(I)} D_i^t(I,\tilde{a})_+} & \text{if } \sum_{\tilde{a}} D_i^t(I, \tilde{a})_+ > 0 \\ \frac{1}{|A(I)|} & \text{otherwise} \end{cases} \tag{7}$$

can be restated in terms of $\hat{D}_i^t(I, a)$.

By definition,

$$\pi_i^{\sigma^t}(I) = \prod_{I' \in I, P(I')=i, a':I' \to I} \sigma_i^t(I', a') \tag{8}$$

Since all σ_i^t have no error by assumption, SD-CFR's recomputation of $\pi_i^{\sigma^t}(I)$ and hence also $\bar{\sigma}_i^T(I, a)$ are exact for any $I \in \mathcal{I}_i$ and all $a \in A(I)$.

To show this for trajectory-sampling SD-CFR, consider a trajectory starting at the tree's root ϕ leading to an information set in $I \in \mathcal{I}_i$. Since σ_i^t can be deduced from $\hat{D}_i^t$ as before, B_i^M can be seen as a buffer of iteration-strategies. Let $f \colon I \to a$ be a function that first chooses a $\sigma_i^t \in B_i^M$, where each σ_i^t is assigned a sampling weight of $t\pi_i^{\sigma}(I)$. f then returns an action sampled from the distribution $\sigma_i^t(I)$. Since f weights strategies like the numerator of the definition

$$\bar{\sigma}_i^T(I, a) = \frac{\sum_{t=1}^T (t\pi_i^{\sigma^t}(I)\sigma_i^t(I, a))}{\sum_{t=1}^T (t\pi_i^{\sigma^t}(I))} \tag{9}$$

executing $f(I)$ is equivalent to sampling directly from $\bar{\sigma}_i^T$.

Note that $\pi_i^{\sigma}(\phi) = 1$ for all σ. Thus, $f(\phi)$ would choose a given $\sigma_i^t \in B_i^M$ with sampling weight t. This is what trajectory-sampling SD-CFR does at ϕ. For each information set I' from ϕ until the end of the trajectory, SD-CFR plays using the same iteration-strategy selected at ϕ. Thus, SD-CFR will reach each information set I with a probability proportional to $\pi_i^{\sigma^t}(I)$ conditional on knowing which iteration-strategy was selected. Combining these facts, we see that the assigned weight of σ_i^t in any I is $t\pi_i^{\sigma^t}(I)$ for any t up to T. It follows that the probability of σ_i^t being the acting policy in any I is

$$\frac{t\pi_i^{\sigma^t}(I)}{\sum_{t'=1}^T (t'\pi_i^{\sigma^{t'}}(I))}$$

Since this is equivalent to the weighting scheme between iteration-strategies in the definition of $\bar{\sigma}_i^T$, trajectory-sampling SD-CFR samples correctly from $\bar{\sigma}_i^T$.

Moreover, because the opponent does not know which σ_i^t is the acting policy, this result also shows that an opponent cannot tell whether the agent is using this sampling method or following an explicitly computed $\bar{\sigma}_i^T$ □

E DEEP CFR PERFORMS WELL ON EARLY ITERATIONS IN SOME GAMES

We conducted experiments searching to investigate the harm caused by the function approximation of $\hat{S}$. We found that in variants of Leduc Hold'em (Southey et al., 2005) with more that 3 ranks and multiple bets, the performance between Deep CFR and SD-CFR was closer. Below we plot the exploitability curves of the early iterations in a variant of Leduc that uses a deck of 12 ranks and allows a maximum of 6 instead of 2 bets per round.

We believe the smaller difference in performance is due to the equilibrium in this game being less sensitive to small differences in action probabilities, while the game is still small enough to see every state often during training. In vanilla Leduc, slight deviations from optimal play give away a lot about one's private information as there are just three distinguishable cards. In contrast, this variant of Leduc, despite having more states, might be less susceptible to approximation error as it has 12 distinguishable cards but similarly simple rules.

For the plot below, we ran Deep CFR and SD-CFR with shared value networks, where all buffers have a capacity of 4 million. On each iteration, data is collected over 8,800 external sampling traversals and a new value network is trained to convergence (1200 updates of batch size 2816), initialized randomly at $t < 2$ and with the weights of the value net from iteration $t - 2$ afterwards. Average-strategy networks are trained to convergence (10000 updates of batch size 5632) from a random initialization. The network architecture used is as Brown et al. (2018a), differing only by the card-branch having 64 units per layer instead of 192.

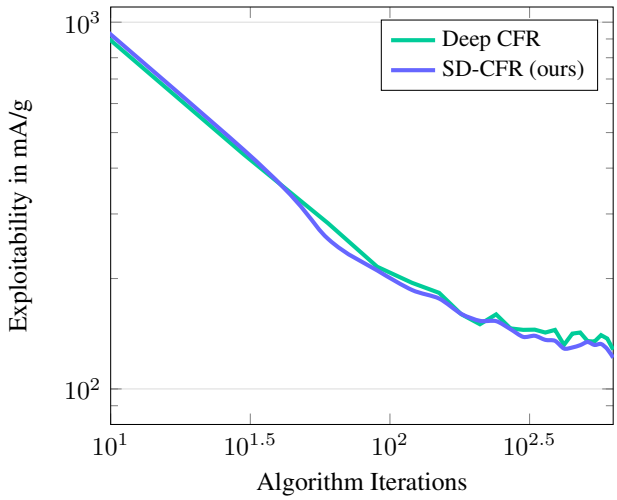


Figure 3: **Exploitability** of Single Deep CFR and Deep CFR averaged over five runs.